

[illegible][illegible]

RRRRRRRR RRRRRRRR RR RR RR RR RRRRRRRR RRRRRRRR RR RR RR RR RR RR RR	PPPPPPPP PPPPPPPP PP PP PP PP PPPPPPPP PPPPPPPP PP PP PP PP PP PP PP	GGGGGGGG GGGGGGGG GG GG GG GG GG GGGGGG GGGGGG GGGGGG GGGGGG GGGGGG	PPPPPPPP PPPPPPPP PP PP PP PP PPPPPPPP PPPPPPPP PP PP PP PP PP PP PP	RRRRRRRR RRRRRRRR RR RR RR RR RRRRRRRR RRRRRRRR RR RR RR RR RR RR RR	IIIIII IIIIII II II II II II II II II II IIIIII IIIIII	NN NN NN NN NNNN NN NN NN NN NN NN NN NN NN	NN NN NN NN NN NN NN NN NN NN NN NN NN NN	TTTTTTTTTT TTTTTTTTTT TT TT TT TT TT TT TT TT TT TT TT TT TT
LL LL LL LL LL LL LL LL LL LL LL LLLLLLLLLL LLLLLLLLLL	IIIIII IIIIII II II II II II II II II II IIIIII IIIIII	SSSSSSSS SSSSSSSS SS SS SS SS SSSSSS SSSSSS SS SS SS SS SSSSSS SSSSSS							


```
1 0001 0 MODULE RPG$PRINT( %TITLE 'Support output to RPG PRINTER files'
2 0002 0 IDENT = '1-003' ! file RPGPRINT.B32 EDIT:LPT1003
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 ++
31 0031 1
32 0032 1 FACILITY: RPGII SUPPORT
33 0033 1
34 0034 1 ABSTRACT:
35 0035 1
36 0036 1 This module contains the RTL routines to handle output to RPG
37 0037 1 PRINTER files for VAX-11 RPGII.
38 0038 1
39 0039 1 ENVIRONMENT: VAX/VMS user mode
40 0040 1
41 0041 1 AUTHOR: Debess Grabazs, CREATION DATE: 20-December-1982
42 0042 1
43 0043 1 MODIFIED BY:
44 0044 1
45 0045 1 1-001 - Original version. DG 20-DEC-82
46 0046 1 1-002 - Added code review comments - most notable code change is the different
47 0047 1 looping techniques for first page forms positioning. DG 26-MAY-83
48 0048 1 1-003 - 1. Add support for overprinting lines. DJB 27-Jun-1983
49 0049 1 2. Only turn on overflow indicator if space or skip after is passed
50 0050 1 the overflow line. LPT 5-Jul-1983
51 0051 1
52 0052 1 --
53 0053 1 !<BLF/PAGE>
```



```
55 0054 1 %SBTTL 'Declarations'
56 0055 1 +
57 0056 1 | PROLOGUE FILE:
58 0057 1 | -
59 0058 1 |
60 0059 1 REQUIRE 'RTLIN:RPGPROLOG';
61 0124 1 |
62 0125 1 |
63 0126 1 | +
64 0127 1 | LINKAGES
65 0128 1 | NONE
66 0129 1 | -
67 0130 1 |
68 0131 1 | +
69 0132 1 | TABLE OF CONTENTS:
70 0133 1 | -
71 0134 1 |
72 0135 1 FORWARD ROUTINE
73 0136 1 RPG$PRINT,
74 0137 1 RPG$TERM_PRINT;
75 0138 1 |
76 0139 1 | +
77 0140 1 | INCLUDE FILES
78 0141 1 | NONE
79 0142 1 | -
80 0143 1 |
81 0144 1 | +
82 0145 1 | MACROS
83 0146 1 | -
84 0147 1 |
85 0148 1 MACRO
86 0149 1 PREFIX = 0,0,8,0%,
87 0150 1 POSTFIX = 0,8,8,0%;
88 0151 1 |
89 0152 1 | +
90 0153 1 | EQUATED SYMBOLS
91 0154 1 | NONE
92 0155 1 | -
93 0156 1 |
94 0157 1 | +
95 0158 1 | EXTERNAL REFERENCES
96 0159 1 | -
97 0160 1 |
98 0161 1 EXTERNAL ROUTINE
99 0162 1 LIB$GET_COMMAND,
100 0163 1 STR$UPCASE;
101 0164 1 |
102 0165 1 EXTERNAL LITERAL
103 0166 1 RPG$_EXTINDOFF;
104 0167 1
```

! Switches, PSECTs, macros,
! linkages and LIBRARYs

! Record header block fields

! Get line from SYS\$COMMAND
! Convert string to uppercase

! File not open error


```
106 0168 1 %SBTTL 'RPG$PRINT - Support output to RPG PRINTER files'
107 0169 1 GLOBAL ROUTINE RPG$PRINT(
108 0170 1     RAB:    REF $RAB_DECL    ! RAB of file to be printed
109 0171 1     ) =
110 0172 1
111 0173 1 ++
112 0174 1
113 0175 1 FUNCTIONAL DESCRIPTION:
114 0176 1
115 0177 1     This routine supports output to RPG PRINTER files. It is called by
116 0178 1     the compiled code once for each write to a PRINTER file.
117 0179 1
118 0180 1     The main function of this routine is to fill in the two-byte
119 0181 1     fixed-length control area which is associated with each record
120 0182 1     and to write the print record to the file. This control area
121 0183 1     contains the spacing controls for a print record. If spacing
122 0184 1     and skipping are both specified for the same line, they are
123 0185 1     performed in the following sequence:
124 0186 1         o Skip before
125 0187 1         o Space before
126 0188 1         o Print the line
127 0189 1         o Skip after
128 0190 1         o Space after.
129 0191 1     The secondary function of this routine is to detect page
130 0192 1     overflow. This occurs only the first time one of the following
131 0193 1     conditions occurs on the current page:
132 0194 1         o A line is printed on the overflow line
133 0195 1         o A line is printed past the overflow line
134 0196 1         o The overflow line is passed during a space operation
135 0197 1         o The overflow line is passed during a skip operation
136 0198 1         (to a line on the current page).
137 0199 1     A special function of this routine is to allow "first page"
138 0200 1     forms positioning. If both RPG$V_CTX_1PFORMS and RPG$V_CTX_FIRST
139 0201 1     are set on, this routine will do the following:
140 0202 1         o PUT the record
141 0203 1         o If RMS returns a failure status, return
142 0204 1         o Issue a message to SYSS$COMMAND to ask the user whether
143 0205 1         forms are positioned correctly
144 0206 1         o Accept "continue" or "retry" as a response
145 0207 1         o If the user responds with "retry", go back to step 1
146 0208 1         o If the user responds with "continue", clear
147 0209 1         RPG$V_CTX_FIRST and return.
148 0210 1
149 0211 1 CALLING SEQUENCE:
150 0212 1
151 0213 1     return_status.wlc.v = RPG$PRINT (rab.rr.r)
152 0214 1
153 0215 1 FORMAL PARAMETERS:
154 0216 1
155 0217 1     rab                                address of the RAB of the file to be
156 0218 1                                         printed.
157 0219 1
158 0220 1 IMPLICIT INPUTS:
159 0221 1
160 0222 1     The implicit inputs for this procedure are contained in the file
161 0223 1     context block. This block is located at a negative offset to the
162 0224 1     RAB. They are defined in RPGDEF.REQ:
```

```
163 0225 1 |
164 0226 1 |   RPG$W_CTX_SPACEB   number of lines to space before printing.
165 0227 1 |
166 0228 1 |   RPG$W_CTX_SPACEA   number of lines to space after printing.
167 0229 1 |
168 0230 1 |   RPG$W_CTX_SKIPB    line number to skip to before printing.
169 0231 1 |
170 0232 1 |   RPG$W_CTX_SKIPA    line number to skip to after printing.
171 0233 1 |
172 0234 1 |   RPG$W_CTX_PFLAGS   flags for print control:
173 0235 1 |
174 0236 1 |       RPG$V_CTX_FIRST TRUE before first write to the file to ensure
175 0237 1 |                       that values get initialized and that the "first
176 0238 1 |                       page" forms positioning takes place, if
177 0239 1 |                       requested, on the first write.
178 0240 1 |
179 0241 1 |       RPG$V_CTX_1PFORMS TRUE when "first page" forms positioning has
180 0242 1 |                       been requested.
181 0243 1 |
182 0244 1 |       RPG$V_CTX_OVLINE TRUE when this is an overflow line.
183 0245 1 |
184 0246 1 |   RPG$W_CTX_LINE     specifies the line number at which the device is
185 0247 1 |                       positioned within the current page body.
186 0248 1 |
187 0249 1 |   RPG$W_CTX_FL       specifies the number of lines in the page body;
188 0250 1 |                       i.e., it specifies the number of lines on the
189 0251 1 |                       logical page that can be written.
190 0252 1 |
191 0253 1 |   RPG$W_CTX_OL       specifies the line number of overflow line.
192 0254 1 |
193 0255 1 |   RPG$A_CTX_OVIND    specifies the address of the overflow indicator
194 0256 1 |                       for this file.
195 0257 1 |
196 0258 1 |   RAB$L_RHB          is the address of the two byte control area
197 0259 1 |                       to contain the print file information. The
198 0260 1 |                       first byte is the "prefix" area, and the second
199 0261 1 |                       byte is the "postfix" area, specifying the
200 0262 1 |                       number of lines to advance before and after the
201 0263 1 |                       record, respectively.
202 0264 1 |
203 0265 1 | IMPLICIT OUTPUTS:
204 0266 1 |
205 0267 1 |     NONE
206 0268 1 |
207 0269 1 | ROUTINE VALUE:
208 0270 1 |
209 0271 1 |     RMS status returned by the PUT operation or RPG$_EXTINDOFF.
210 0272 1 |
211 0273 1 | SIDE EFFECTS:
212 0274 1 |
213 0275 1 |     A PUT to the lineage file is performed.
214 0276 1 |
215 0277 1 | --
```



```
217 0278 2 BEGIN
218 0279 2
219 0280 2 LITERAL
220 0281 2 SET_ON = 1,
221 0282 2 SET_OFF = 0,
222 0283 2 SET_OFF_OVERFLOW = %X'FFFEFEFE',
223 0284 2
224 0285 2
225 0286 2
226 0287 2 SET_ON_OVERFLOW = %X'00010101';
227 0288 2
228 0289 2
229 0290 2
230 0291 2
231 0292 2 LOCAL
232 0293 2 ADV_LINES,
233 0294 2 LINE_FLAG: WORD,
234 0295 2
235 0296 2
236 0297 2 RET_STATUS,
237 0298 2 RHB: REF BLOCK[,BYTE];
238 0299 2
239 0300 2 BIND
240 0301 2 FCB = RAB : REF BLOCK [,BYTE];
241 0302 2
242 0303 2 BUILTIN
243 0304 2 TESTBITSC;
244 0305 2
245 0306 2
246 0307 2
247 0308 2
248 0309 2
249 0310 2
250 0311 2 IF .RAB[RAB$W_ISI] EQL 0
251 0312 2 THEN
252 0313 2 RETURN RPG$_EXTINDOFF;
253 0314 2
254 0315 2
255 0316 2
256 0317 2
257 0318 2
258 0319 2
259 0320 2
260 0321 2
261 0322 2
262 0323 2
263 0324 2
264 0325 2
265 0326 2
266 0327 2
267 0328 2
268 0329 2
269 0330 2
270 0331 2
271 0332 2
272 0333 2
273 0334 2

! Context bit is set on if it equals one
! Context bit is set off if it equals zero
! Overflow indicator is set off if the low bit
! in the byte pointed to by RPG$A_CTX_OVIND and
! the low bit in each of the following two bytes
! is cleared
! Overflow indicator is set on if the low bit
! in the byte pointed to by RPG$A_CTX_OVIND and
! the low bit in each of the following two bytes
! is set (note this mask is the NOT of SET_OFF_OVERFLOW)

! Number of lines to advance
! Flag to ensure print page
! overflow occurs only once
! per page
! Return status
! Record header block

! File context block

! RPG$PRINT should not cause access violations. Since RPG$PRINT is called
! before the associated $PUT, the RAB may be invalid. Validate the RAB
! by checking that RAB$W_ISI is non-zero.

! IF .RAB[RAB$W_ISI] EQL 0
! THEN
! RETURN RPG$_EXTINDOFF;

! Initialization
! LINE_FLAG = .FCB[RPG$W_CTX_LINE];
! RHB = .RAB[RAB$L_RHB];
! RHB[PREFIX] = 0;
! RHB[POSTFIX] = 0;

! Set overflow flag
! Point to control area
! Clear control area

! Process skipping and spacing before the print
! IF .FCB[RPG$W_CTX_SKIPB] GTR 0
! THEN
! BEGIN
! ! Skip before
! SKIP BEFORE indicated
! ADV_LINES = .FCB[RPG$W_CTX_SKIPB] - .FCB[RPG$W_CTX_LINE];
! IF .ADV_LINES NEQ 0
! ! Number of lines to advance
! ! Make sure SKIP TO line
! ! is not current line
```



```
274 0335 3      THEN
275 0336 4          BEGIN
276 0337 4              ! New line
277 0338 4          FCB[RPG$W_CTX_LINE] = .FCB[RPG$W_CTX_SKIPB];
278 0339 4              ! Update current line
279 0340 4          IF .ADV_LINES LSS 0
280 0341 5      THEN
281 0342 5          BEGIN
282 0343 5              ! SKIP BEFORE will cause advance to a new page
283 0344 5          RHB[PREFIX] = .FCB[RPG$W_CTX_FL] + .ADV_LINES;
284 0345 5              ! Set prefix in control area
285 0346 5          LINE_FLAG = 0;
286 0347 5              ! Flag reset for new page
287 0348 5          FCB[RPG$V_CTX_OVPEND] = SET OFF;
288 0349 5              ! 1-003 Flag reset for new page
289 0350 5          IF .FCB[RPG$V_CTX_OVLINE] NEQ SET_ON
290 0351 5      THEN
291 0352 5          .FCB[RPG$A_CTX_OVIND] = ..FCB[RPG$A_CTX_OVIND] AND SET OFF_OVERFLOW;
292 0353 5              ! Set off the overflow indicator
293 0354 5      END
294 0355 4      ELSE
295 0356 4          ! SKIP TO line will be on the same page
296 0357 4          RHB[PREFIX] = .ADV_LINES;
297 0358 4              ! Set prefix in control area
298 0359 4              ! Set on the overflow indicator
299 0360 4      END;
300 0361 3          ! New line
301 0362 3      END;
302 0363 2          ! Skip before
303 0364 2      IF .FCB[RPG$W_CTX_SPACEB] GTR 0
304 0365 2      THEN
305 0366 2          BEGIN
306 0367 2              ! SPACE BEFORE indicated
307 0368 2          FCB[RPG$W_CTX_LINE] = .FCB[RPG$W_CTX_LINE] + .FCB[RPG$W_CTX_SPACEB];
308 0369 2              ! Update current line
309 0370 2          RHB[PREFIX] = .RHB[PREFIX] + .FCB[RPG$W_CTX_SPACEB];
310 0371 2              ! Adjust prefix in control area
311 0372 2          END
312 0373 2              ! 1-003
313 0374 2          ELSE
314 0375 2              ! 1-003
315 0376 2              ! 1-003
316 0377 2              ! 1-003
317 0378 2              ! If the skip caused no advance, then we are going to print on
318 0379 2              ! the same line as the previous PUT, so we need the specify CR
319 0380 2              ! in the prefix area to get overprinting.
320 0381 2              ! 1-003
321 0382 2          IF .RHB[PREFIX] EQL 0
322 0383 2              ! 1-003
323 0384 2          THEN
324 0385 2              RHB[PREFIX] = %X'8D';
325 0386 2              ! 1-003
326 0387 2          ! Check for line being printed on or past the overflow line
327 0388 2      IF .FCB[RPG$W_CTX_LINE] GEQ .FCB[RPG$W_CTX_OL]
328 0389 2      THEN
329 0390 2          IF (.LINE_FLAG LSS .RAB[RPG$W_CTX_OL]) OR
330 0391 2              ! First time on this page?
```



```
331 0392 3      (.FCB[RPG$V_CTX_OVPEND] EQL SET_ON)      ! 1-003 Was an overflow pending?
332 0393 3      THEN
333 0394 3      BEGIN
334 0395 3      .FCB[RPG$A_CTX_OVIND] = ..FCB[RPG$A_CTX_OVIND] OR SET_ON_OVERFLOW;
335 0396 3      FCB[RPG$V_CTX_OVPEND] = SET_OFF;      ! Yes, set on the overflow indicator
336 0397 3      ! 1-003
337 0398 3      END;
338 0399 3
339 0400 3
340 0401 3
341 0402 3      ! Check for current line being on new page
342 0403 3      IF .FCB[RPG$W_CTX_LINE] GTR .FCB[RPG$W_CTX_FL]
343 0404 3      THEN
344 0405 3      FCB[RPG$W_CTX_LINE] = .FCB[RPG$W_CTX_LINE] - .FCB[RPG$W_CTX_FL]; ! Adjust current line to reflect
345 0406 3      ! new page
346 0407 3
347 0408 3
348 0409 3
349 0410 3      ! Process skipping and spacing after the print
350 0411 3      IF .FCB[RPG$W_CTX_SKIPA] GTR 0
351 0412 3      THEN
352 0413 3      BEGIN      ! Skip after
353 0414 3
354 0415 3      SKIP AFTER indicated
355 0416 3
356 0417 3      ADV_LINES = .FCB[RPG$W_CTX_SKIPA] - .FCB[RPG$W_CTX_LINE];      ! Number of lines to advance
357 0418 3      IF .ADV_LINES NEQ 0      ! Make sure SKIP TO line
358 0419 3      THEN      ! is not current line
359 0420 3      BEGIN      ! New line
360 0421 3      FCB[RPG$W_CTX_LINE] = .FCB[RPG$W_CTX_SKIPA];      ! Update current line
361 0422 3      IF .ADV_LINES LSS 0
362 0423 3      THEN
363 0424 3      BEGIN
364 0425 3      SKIP AFTER will cause advance to a new page
365 0426 3
366 0427 3      RHB[POSTFIX] = .FCB[RPG$W_CTX_FL] + .ADV_LINES;      ! Set postfix in control area
367 0428 3      LINE_FLAG = 0;      ! Reset flag for new page
368 0429 3      IF .FCB[RPG$V_CTX_OVLINE] NEQ SET_ON
369 0430 3      THEN
370 0431 3      .FCB[RPG$A_CTX_OVIND] = ..FCB[RPG$A_CTX_OVIND] AND SET_OFF_OVERFLOW;
371 0432 3      ! Set off the overflow indicator
372 0433 3      END
373 0434 3      ELSE
374 0435 3      SKIP AFTER line will be on the same page
375 0436 3      RHB[POSTFIX] = .ADV_LINES;      ! Set postfix in control area
376 0437 3      END;      ! New line
377 0438 3      END;      ! Skip after
378 0439 3
379 0440 3
380 0441 3
381 0442 3
382 0443 3
383 0444 3
384 0445 3
385 0446 3
386 0447 3
387 0448 3
```



```
388 0449 2 IF .FCB[RPG$W_CTX_SPACEA] GTR 0
389 0450 THEN
390 0451 BEGIN
391 0452
392 0453 SPACE AFTER indicated
393 0454
394 0455 FCB[RPG$W_CTX_LINE] = .FCB[RPG$W_CTX_LINE] + .FCB[RPG$W_CTX_SPACEA];
395 0456
396 0457 RHB[POSTFIX] = .RHB[POSTFIX] + .FCB[RPG$W_CTX_SPACEA];
397 0458
398 0459 END;
399 0460
400 0461
401 0462
402 0463
403 0464 IF (.FCB[RPG$W_CTX_LINE] GTR .FCB[RPG$W_CTX_OL]) AND
404 0465 (.LINE_FLAG LSS .FCB[RPG$W_CTX_OL])
405 0466 THEN
406 0467 .FCB[RPG$A_CTX_OVIND] = ..FCB[RPG$A_CTX_OVIND] OR SET_ON_OVERFLOW
407 0468
408 0469 ELSE
409 0470 IF (.FCB[RPG$W_CTX_LINE] EQL .FCB[RPG$W_CTX_OL]) AND
410 0471 (.LINE_FLAG LSS .FCB[RPG$W_CTX_OL])
411 0472 THEN
412 0473 FCB[RPG$V_CTX_OVPEND] = SET_ON;
413 0474
414 0475
415 0476
416 0477
417 0478
418 0479 IF .FCB[RPG$W_CTX_LINE] GTR .FCB[RPG$W_CTX_FL]
419 0480 THEN
420 0481 FCB[RPG$W_CTX_LINE] = .FCB[RPG$W_CTX_LINE] - .FCB[RPG$W_CTX_FL];
421 0482
422 0483
423 0484
424 0485
425 0486
426 0487
427 0488
428 0489
429 0490
430 0491
431 0492
432 0493
433 0494
434 0495
435 0496
436 0497
437 0498
438 0499
439 0500
440 0501
441 0502
442 0503
443 0504
444 0505

      ! First page forms positioning
      LOCAL
      GET STATUS,
      PROMPT_DESC: BLOCK[8,BYTE],
      RESP_DESC: BLOCK[8,BYTE],
      RESP_BUF: VECTOR[10,BYTE];
      LITERAL
      TRUE = 1,
      MIN_RESP_LEN = %CHARCOUNT('xxx');

      ! Return status from LIB$GET_COMMAND
      ! Local descriptor for prompt
      ! Local descriptor for response
      ! Buffer for response
      ! Minimum acceptable length of
      ! response to LIB$GET_COMMAND
```



```
445 0506 LABEL
446 0507 OUTER_LOOP;
447 0508
448 0509 BIND
449 0510
450 0511 | NOTE - PROMPT must come directly before RET for the prompt
451 0512 | string length to be calculated correctly
452 0513
453 0514 PROMPT = UPLIT (' Is forms positioning correct? Yes, type CONTINUE, No, type RETRY: '),
454 0515 RET = UPLIT ('RET'),
455 0516 CON = UPLIT ('CON');
456 0517
457 0518 |
458 0519 | 'First page' forms positioning indicated
459 0520
460 0521 PROMPT_DESC[DSC$W_LENGTH] = CH$DIFF (RET, PROMPT);
461 0522 PROMPT_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
462 0523 PROMPT_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
463 0524 PROMPT_DESC[DSC$A_POINTER] = PROMPT;
464 0525 RESP_DESC[DSC$W_LENGTH] = %ALLOCATION (RESP_BUF);
465 0526 RESP_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
466 0527 RESP_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
467 0528 RESP_DESC[DSC$A_POINTER] = RESP_BUF;
468 0529
469 0530 4 OUTER_LOOP: BEGIN
470 0531 4
471 0532 4 WHILE TRUE DO
472 0533 4 BEGIN ! Retry loop
473 0534 5
474 0535 5 | PUT the record
475 0536 5
476 0537 5 RET STATUS = $PUT(RAB = .RAB); ! Put out the record
477 0538 6 IF NOT (.RET_STATUS)
478 0539 5 THEN
479 0540 6 BEGIN
480 0541 6 |
481 0542 6 | Error on PUT, return
482 0543 6
483 0544 6 FCB[RPG$V CTX FIRST] = SET_ON; ! Reset FIRST bit
484 0545 6 RETURN .RET_STATUS;
485 0546 6
486 0547 5 END;
487 0548 5
488 0549 5 | Issue a message to SYS$COMMAND to ask the user
489 0550 5 | whether forms are positioned correctly.
490 0551 5 | If response is neither RET(RY) or CON(TINUE),
491 0552 5 | prompt again.
492 0553 5 | If response is RETRY, go thru outer loop again.
493 0554 5
494 0555 5 WHILE TRUE DO
495 0556 6 BEGIN
496 0557 6
497 0558 6 DO ! Prompt for response until user types
498 0559 6 ! RET(RY) or CON(TINUE)
499 0560 6
500 0561 6 GET STATUS = LIB$GET_COMMAND(RESP_DESC, PROMPT_DESC)
501 0562 6 UNTIL .GET_STATUS;
```

```

502 0563 6 STR$UPCASE (RESP_DESC, RESP_DESC);
503 0564 6 IF CH$EQL (MIN_RESP_LEN, RESP_BUF, MIN_RESP_LEN, CON)
504 0565 6 THEN
505 0566 6 LEAVE OUTER_LOOP;
506 0567 6 IF CH$EQL (MIN_RESP_LEN, RESP_BUF, MIN_RESP_LEN, RET)
507 0568 6 THEN
508 0569 6 EXITLOOP;
509 0570 6
510 0571 5 END;
511 0572 5
512 0573 4 END; ! Retry loop
513 0574 4
514 0575 3 END; ! Outer loop
515 0576 3 RETURN .RET_STATUS; ! Return status from PUT
516 0577 3
517 0578 2 END; ! First page forms positioning
518 0579 2
519 0580 2 !
520 0581 2 ! When not special-casing, will get here.
521 0582 2
522 0583 2 RETURN $PUT(RAB = .RAB); ! PUT out the record and
523 0584 2 ! return the RMS status
524 0585 2
525 0586 1 END;
```

```

74 69 73 6F 70 20 73 6D 72 6F 66 20 73 49 20 00000 P.AAA: .TITLE RPG$PRINT Support output to RPG PRINTER files
3F 74 63 65 72 72 6F 63 20 67 6E 69 6E 6F 69 0000F .IDENT \1-003\
20 2C 6F 4E 20 2C 45 55 4E 49 54 4E 4F 43 20 0001E .PSECT _RPG$CODE,NOWRT, SHR, PIC,2
00 20 3A 59 52 54 45 52 20 65 70 79 74 00028 .ASCII \ Is forms positioning correct? Yes, type\
00 00 54 45 52 00044 P.AAB: .ASCII \RET\<0>
00 4E 4F 43 00048 P.AAC: .ASCII \CON\<0>
                                PROMPT= P.AAA
                                RET= P.AAB
                                CON= P.AAC
                                .EXTRN LIB$GET_COMMAND
                                .EXTRN STR$UPCASE, RPG$_EXTINDOFF
                                .EXTRN SYS$PUT
                                .ENTRY RPG$PRINT, Save R2,R3,R4,R5,R6,R7
57 0000C000G 00 00FC 00000 MOVAB SYS$PUT, R7
5E 1C C2 00009 SUBL2 #28, SP
54 04 AC D0 0000C MOVL RAB, R4
02 A4 B5 00010 TSTW 2(R4)
08 12 00013 BNEQ 1$
50 00000000G 8F D0 00015 MOVL #RPG$_EXTINDOFF, R0
04 0001C RET
50 EE A4 9E 0001D 1$: MOVAB -18(R4), R0
55 60 B0 00021 MOVW (R0), LINE_FLAG
51 2C A4 D0 00024 MOVL 44(R4), RHB
                                : 0169
                                : 0311
                                : 0313
                                : 0318
                                : 0319
```


			61	B4	00028	CLRW	(RHB)	0320
		53	EA	A4	3C 0002A	MOVZWL	-22(R4), R3	0326
				2D	15 0002E	BLEQ	3\$	
		52		60	3C 00030	MOVZWL	(R0), ADV_LINES	0332
52		53		52	C3 00033	SUBL3	ADV_LINES, R3, ADV_LINES	
				24	13 00037	BEQL	3\$	0333
		60		53	B0 00039	MOVW	R3, (R0)	0338
				52	D5 0003C	TSTL	ADV_LINES	0339
				1A	18 0003E	BGEQ	2\$	
61	F2	A4		52	81 00040	ADDB3	ADV_LINES, -14(R4), (RHB)	0345
				55	B4 00045	CLRW	LINE_FLAG	0346
	EC	A4		08	8A 00047	BICB2	#8, -20(R4)	0347
0D	EC	A4		02	E0 0004B	BBS	#2, -20(R4), 3\$	0348
	F4	B4	00010101	8F	CA 00050	BICL2	#65793, a-12(R4)	0350
				03	11 00058	BRB	3\$	0339
		61		52	90 0005A	MOVB	ADV_LINES, (RHB)	0358
			E6	A4	B5 0005D	TSTW	-26(R4)	0365
				0A	13 00060	BEQL	4\$	
		60		A4	A0 00062	ADDW2	-26(R4), (R0)	0371
		61		A4	80 00066	ADDB2	-26(R4), (RHB)	0373
			E6	08	11 0006A	BRB	5\$	0365
				61	95 0006C	TSTB	(RHB)	0382
				04	12 0006E	BNEQ	5\$	
		61	8D	8F	90 00070	MOVB	#-115, (RHB)	0384
	F0	A4		60	B1 00074	CMPW	(R0), -16(R4)	0389
				17	1F 00078	BLSSU	7\$	
	F0	A4		55	B1 0007A	CMPW	LINE_FLAG, -16(R4)	0391
				05	1F 0007E	BLSSU	6\$	
0C	EC	A4		03	E1 00080	BBC	#3, -20(R4), 7\$	0392
	F4	B4	00010101	8F	C8 00085	BISL2	#65793, a-12(R4)	0396
	EC	A4		08	8A 0008D	BICB2	#8, -20(R4)	0398
	F2	A4		60	B1 00091	CMPW	(R0), -14(R4)	0405
				04	1B 00095	BLEQU	8\$	
		60	F2	A4	A2 00097	SUBW2	-14(R4), (R0)	0407
		53	E8	A4	3C 0009B	MOVZWL	-24(R4), R3	0413
				2B	15 0009F	BLEQ	10\$	
		52		60	3C 000A1	MOVZWL	(R0), ADV_LINES	0419
52		53		52	C3 000A4	SUBL3	ADV_LINES, R3, ADV_LINES	
				22	13 000A8	BEQL	10\$	0420
		60		53	B0 000AA	MOVW	R3, (R0)	0424
				52	D5 000AD	TSTL	ADV_LINES	0425
				17	18 000AF	BGEQ	9\$	
01	A1	F2	A4	52	81 000B1	ADDB3	ADV_LINES, -14(R4), 1(RHB)	0431
				55	B4 000B7	CLRW	LINE_FLAG	0432
0E	EC	A4		02	E0 000B9	BBS	#2, -20(R4), 10\$	0433
	F4	B4	00010101	8F	CA 000BE	BICL2	#65793, a-12(R4)	0435
				04	11 000C6	BRB	10\$	0425
		01	A1	52	90 000C8	MOVB	ADV_LINES, 1(RHB)	0443
			E4	A4	B5 000CC	TSTW	-28(R4)	0449
				09	13 000CF	BEQL	11\$	
		60	E4	A4	A0 000D1	ADDW2	-28(R4), (R0)	0455
	01	A1	E4	A4	80 000D5	ADDB2	-28(R4), 1(RHB)	0457
	F0	A4		60	B1 000DA	CMPW	(R0), -16(R4)	0464
				10	1B 000DE	BLEQU	12\$	
	F0	A4		55	B1 000E0	CMPW	LINE_FLAG, -16(R4)	0465
				0A	1E 000E4	BGEQU	12\$	
	F4	B4	00010101	8F	C8 000E6	BISL2	#65793, a-12(R4)	0467

				10	11	000EE		BRB	13\$		
	F0	A4		60	B1	000F0	12\$:	CMPW	(R0), -16(R4)	:	0470
				0A	12	000F4		BNEQ	13\$:	
	F0	A4		55	B1	000F6		CMPW	LINE_FLAG, -16(R4)	:	0471
				04	1E	000FA		BGEQU	13\$:	
	EC	A4		08	88	000FC		BISB2	#8, -20(R4)	:	0473
	F2	A4		60	B1	00100	13\$:	CMPW	(R0), -14(R4)	:	0478
				04	1B	00104		BLEQU	14\$:	
		60	F2	A4	A2	00106		SUBW2	-14(R4), (R0)	:	0480
66	EC	A4		00	E5	0010A	14\$:	BBCC	#0, -20(R4), 18\$:	0488
61	EC	A4		01	E1	0010F		BBC	#1, -20(R4), 18\$:	0490
	14	AE	010E0044	8F	D0	00114		MOVL	#17694788, PROMPT_DESC	:	0521
	18	AE	FE94	CF	9E	0011C		MOVAB	PROMPT, PROMPT_DESC+4	:	0524
	OC	AE	010E000A	8F	D0	00122		MOVL	#17694730, RESP_DESC	:	0525
	10	AE		6E	9E	0012A		MOVAB	RESP_BUF, RESP_DESC+4	:	0528
				54	DD	0012E	15\$:	PUSHL	R4	:	0537
		67		01	FB	00130		CALLS	#1, SYSSPUT	:	
		55		50	D0	00133		MOVL	R0, RET_STATUS	:	
		06		55	E8	00136		BLBS	RET_STATUS, 16\$:	0538
	EC	A4		01	88	00139		BISB2	#1, -20(R4)	:	0544
				32	11	0013D		BRB	17\$:	0545
			14	AE	9F	0013F	16\$:	PUSHAB	PROMPT_DESC	:	0561
			10	AE	9F	00142		PUSHAB	RESP_DESC	:	
	00000000G	00		02	FB	00145		CALLS	#2, LIB\$GET_COMMAND	:	
		56		50	D0	0014C		MOVL	R0, GET_STATUS	:	
		ED		56	E9	0014F		BLBC	GET_STATUS, 16\$:	0562
			OC	AE	9F	00152		PUSHAB	RESP_DESC	:	0563
			10	AE	9F	00155		PUSHAB	RESP_DESC	:	
	00000000G	00		02	FB	00158		CALLS	#2, STR\$UPCASE	:	
FE97	CF	6E		03	29	0015F		CMPC3	#3, RESP_BUF, CON	:	0564
				0A	13	00165		BEQL	17\$:	
FE8B	CF	6E		03	29	00167		CMPC3	#3, RESP_BUF, RET	:	0567
				D0	12	0016D		BNEQ	16\$:	
				BD	11	0016F		BRB	15\$:	0569
		50		55	D0	00171	17\$:	MOVL	RET_STATUS, R0	:	0576
					04	00174		RET		:	
				54	DD	00175	18\$:	PUSHL	R4	:	0583
		67		01	FB	00177		CALLS	#1, SYSSPUT	:	
				04	0017A			RET		:	0586

; Routine Size: 379 bytes, Routine Base: _RPG\$CODE + 004C


```

: 527 0587 1 %SBTTL 'RPG$TERM_PRINT - Finish logical page'
: 528 0588 1 GLOBAL ROUTINE RPG$TERM_PRINT(
: 529 0589 1     RAB: REF $RAB_DECL ! RAB of file to be printed
: 530 0590 1     )=
: 531 0591 1
: 532 0592 1 ++
: 533 0593 1
: 534 0594 1 FUNCTIONAL DESCRIPTION:
: 535 0595 1
: 536 0596 1     This routine is called to advance the number of lines needed to
: 537 0597 1     finish out the logical page before the actual CLOSE is done.
: 538 0598 1
: 539 0599 1
: 540 0600 1 CALLING SEQUENCE:
: 541 0601 1
: 542 0602 1     return_status.wlc.v = RPG$TERM_PRINT (rab.rr.r)
: 543 0603 1
: 544 0604 1 FORMAL PARAMETERS:
: 545 0605 1
: 546 0606 1     rab                                address of the RAB of the file to be
: 547 0607 1                                printed.
: 548 0608 1
: 549 0609 1 IMPLICIT INPUTS:
: 550 0610 1
: 551 0611 1     RPG$W_CTX_FL                        specifies the number of lines in the page body;
: 552 0612 1                                i.e., it specifies the number of lines on the
: 553 0613 1                                logical page that can be written.
: 554 0614 1
: 555 0615 1     RPG$W_CTX_LINE                       specifies the line number at which the device is
: 556 0616 1                                positioned within the current page body.
: 557 0617 1
: 558 0618 1
: 559 0619 1 IMPLICIT OUTPUTS:
: 560 0620 1
: 561 0621 1     A PUT to the lineage file is performed
: 562 0622 1
: 563 0623 1 ROUTINE VALUE:
: 564 0624 1
: 565 0625 1     RMS status returned by the PUT operation or SS$_NORMAL if
: 566 0626 1     nothing needs to be done by this routine.
: 567 0627 1
: 568 0628 1 SIDE EFFECTS:
: 569 0629 1
: 570 0630 1     NONE
: 571 0631 1
: 572 0632 1 --
```



```
: 574      0633 2  BEGIN
: 575      0634 2
: 576      0635 2  LITERAL
: 577      0636 2    SET_ON = 1;
: 578      0637 2
: 579      0638 2  LOCAL
: 580      0639 2    RHB : REF BLOCK [,BYTE];          ! Record header block
: 581      0640 2
: 582      0641 2  BIND
: 583      0642 2    FCB = RAB : REF BLOCK [,BYTE];      ! File context block
: 584      0643 2
: 585      0644 2
: 586      0645 2    !
: 587      0646 2    ! RPG$TERM_PRINT should not cause access violations. Since it WILL
: 588      0647 2    ! be called before the associated SYS$CLOSE, the RAB may be invalid.
: 589      0648 2    ! Validate the RAB by checking that RAB$W_ISI is non-zero.
: 590      0649 2
: 591      0650 2  IF .RAB[RAB$W_ISI] EQL 0
: 592      0651 2  THEN
: 593      0652 2    RETURN RPG$_EXTINDOFF;
: 594      0653 2
: 595      0654 2    !
: 596      0655 2    ! If no WRITE has ever been done for this file, then no adjustment
: 597      0656 2    ! need be done at CLOSE time. Note that the flag bit is checked
: 598      0657 2    ! but not cleared; if it is set, we will not be doing a WRITE either.
: 599      0658 2
: 600      0659 2  IF .FCB[RPG$V_CTX_FIRST] EQL SET_ON
: 601      0660 2  THEN
: 602      0661 2    RETURN SS$_NORMAL;
: 603      0662 2
: 604      0663 2    !
: 605      0664 2    ! Figure out how many lines left to fill out the page
: 606      0665 2
: 607      0666 2  RHB = .RAB[RAB$L_RHB];
: 608      0667 2  RHB[PREFIX] = .FCB[RPG$W_CTX_FL] - .FCB[RPG$W_CTX_LINE] + 1;
: 609      0668 2
: 610      0669 2    !
: 611      0670 2    ! Make sure that there is something to advance.
: 612      0671 2
: 613      0672 2  IF .RHB[PREFIX] EQL 0
: 614      0673 2  THEN
: 615      0674 2    RETURN SS$_NORMAL;
: 616      0675 2
: 617      0676 2    !
: 618      0677 2    ! The actual WRITE is done by PUTting a record of 0 length with appropriate
: 619      0678 2    ! advance in the PRN control fields.
: 620      0679 2
: 621      0680 2  RAB[RAB$W_RSZ] = 0;
: 622      0681 2  RHB[POSTFIX] = 0;
: 623      0682 2
: 624      0683 2  RETURN $PUT(RAB = .RAB);
: 625      0684 1  END;
```


			000C 00000	.ENTRY	RPG\$TERM_PRINT, Save R2,R3		0588
50	04	AC	D0 00002	MOVL	RAB, R0		0649
	02	A0	B5 00006	TSTW	2(R0)		
		08	12 00009	BNEQ	1\$		
50	00000000G	8F	D0 0000B	MOVL	#RPG\$_EXTINDOFF, R0		0651
			04 00012	RET			
15	EC	A0	E8 00013	BLBS	-20(R0), 2\$		0658
52	2C	A0	D0 00017	MOVL	44(R0), RHB		0665
51	F2	A0	3C 0001B	MOVZWL	-14(R0), R1		0666
53	EE	A0	3C 0001F	MOVZWL	-18(R0), R3		
51		53	C2 00023	SUBL2	R3, R1		
62		01	81 00026	ADDB3	#1, R1, (RHB)		
51		04	12 0002A	BNEQ	3\$		0671
50		01	D0 0002C	MOVL	#1, R0		0673
			04 0002F	RET			
	22	A0	B4 00030	CLRW	34(R0)		0679
	01	A2	94 00033	CLRB	1(RHB)		0680
		50	DD 00036	PUSHL	R0		0682
00000000G 00		01	FB 00038	CALLS	#1, SYS\$PUT		
			04 0003F	RET			0684

; Routine Size: 64 bytes, Routine Base: _RPG\$CODE + 01C7

: 626 0685 1
: 627 0686 0 END ELUDGM

PSECT SUMMARY

Name	Bytes	Attributes
_RPG\$CODE	519	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	16	0	581	00:01.0
_\$255\$DUA28:[RPGRTL.OBJ]RPGLIB.L32;1	54	12	22	9	00:00.1

COMMAND QUALIFIERS

RPG\$PRINT
1-003

Support output to RPG PRINTER files
RPG\$TERM_PRINT - Finish logical page

D 4
16-Sep-1984 02:18:04
14-Sep-1984 13:04:24

VAX-11 Bliss-32 V4.0-742
[RPGRTL.SRC]RPGPRINT.B32;1

Page 16
(6)

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:RPGPRINT/OBJ=OBJ\$:RPGPRINT MSRC\$:RPGPRINT/UPDATE=(ENH\$:RPGPRINT)
: Size: 443 code + 76 data bytes
: Run Time: 00:13.7
: Elapsed Time: 00:46.1
: Lines/CPU Min: 3000
: Lexemes/CPU-Min: 21161
: Memory Used: 189 pages
: Compilation Complete

0332 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

